Asynchronous Circuits, Communicating Processes, and Muller Automaton*

Roman R. Redziejowski

Abstract

A finite-state model of communicating processes is presented alongside with the Muller-Bartky model of asynchronous circuits. The purpose of such presentation is to expose a close similarity of these models, at a level of abstraction much lower than the Muller automaton.

1 Introduction

In a pioneering paper [6] from 1959, Muller and Bartky used a finite-state model to study race conditions in asynchronous circuits. To analyze certain type of behavior, they had to consider infinite sequences of transitions in the model. An essential part of the model was an assumption, later known as the "finite delay property", that reflected the unknown delays introduced by circuit components. The problem was reduced to studying behavior of a nondeterministic, inputless automaton that could undergo finite or infinite sequences of transitions. The analysis of this model led Muller in [5] to a special type of automaton that generated finite and infinite words over a finite alphabet. Automata of this kind, known as "Muller automata", became subsequently popular in a different context, as deterministic recognizers of ω-regular languages (see, for example, [3, 7, 9]).

In a pioneering report [1] from 1965 (later published as [2]), Dijkstra described the phenomenon of dynamic blocking in a system of communicating processes. A natural way of studying this phenomenon was to consider infinite behavior of the system. Dijkstra's assumption of processes having "unknown, but non-zero speed" was very close to the finite delay property. The idea of infinite behavior restricted by finite delay property, or by the related property of "fairness", became subsequently a standard ingredient in almost every model of concurrent processes. The Muller automaton followed in a natural way. (One of the early applications of this idea was the present author's report [8].)

In the Muller-Bartky model, several variables change in a mutually dependent way according to certain laws. The variables have a tendency to change that depends on current values of the variables. One can see there a remote analogy to continuous systems described by differential equations, with tendencies corresponding to derivatives. The important difference (besides that of the model being discrete) is that the tendencies are specified in an incomplete way.

The purpose of this paper is to show how asynchronous circuits and concurrent processes can be described in a common way according to this view. We present the Muller-Bartky model and a finite-state model of communicating processes using the same formalism. The analysis, and the derivation of Muller automaton, are then carried out in a generalized form that applies to both.

2 Muller-Bartky model of asynchronous circuits

We have a circuit consisting of $n > 1$ components $C_1, C_2, \ldots, C_n$, and study its behavior after all inputs have been fixed. For $1 \leq k \leq n$, the state of C_k (output voltage, relay position, etc.) can assume values from a finite set S_k. The state of the system is described by an n-tuple $s = (s_1, s_2, \ldots, s_n)$ where $s_k \in S_k$ is the state of C_k for $1 \leq k \leq n$. We denote the set of all these n-tuples by S.

Each set S_k is ordered. For example, in a binary circuit $S_k = \{0, 1\}$ with $0 < 1$; in general, $S_k = \{0, 1, \ldots, j_k\}$ with $0 < 1 < ... < j_k$.

*Appeared in *Fundamenta Informaticae 61* (2004) 47–59.

The switching properties of the circuit are described by n functions $f_k : S \to S_k$. In the state $s \in S$ of the circuit, component C_k tends to change its state to $f_k(s)$ – unless it already is in this state. The components operate with unknown (and usually unequal) delays. When they eventually change state, not necessarily all of them will do it, and perhaps not all the way towards $f_k(s)$. We only know that the new state of each component must be somewhere in the closed interval between the old value and $f_k(s)$. In other words, if the state s of the circuit changes, it can only change to a state from the set

$$\nu(s) = \{s' \in S \mid s\, R\, s' \ \wedge\ s \neq s'\},$$

where $s\, R\, s'$ means that one of the following holds for $1 \leq k \leq n$:

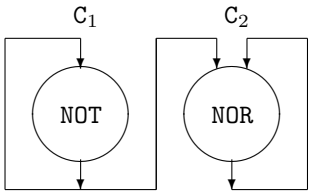
$$s_k \leq s'_k \leq f_k(s), \text{ or}$$
$$s_k \geq s'_k \geq f_k(s).$$

We know further that if a component has tendency to change state, it will do so in a finite, but unspecified, time; this is the finite delay property mentioned before. For $s \in S$ and $1 \leq k \leq n$, define:

$$\partial_k(s) = - \ \text{ if } \ f_k(s) < s_k,$$
$$\partial_k(s) = \ 0 \ \text{ if } \ f_k(s) = s_k,$$
$$\partial_k(s) = + \ \text{ if } \ f_k(s) > s_k.$$

The value of $\partial_k(s)$ expresses the tendency of S_k to change state with circuit in state s, with "$-$" meaning a tendency to decrease, "$+$" a tendency to increase, and "0" no tendency to change. A precise formulation of finite delay property is: if $\partial_k(s) \neq 0$ for some s and k, either the value of ∂_k or the state of S_k must change within a finite time. This takes care of situations where the tendency is intermittent or disppears.

Example 1



$S_1 = S_2 = \{0, 1\}, \ \ 0 < 1$

$f_1(s) = \overline{s_1}$

$f_2(s) = \overline{s_1} \ \wedge \ \overline{s_2}$

Figure 1: An asynchronous circuit.

s	$\nu(s)$	$\partial(s)$
00	{01,10,11}	$++$
01	{00,10,11}	$+-$
10	{00}	-0
11	{00,01,10}	$--$

Figure 2: Functions ν and ∂ for the circuit of Figure 1.

A binary circuit of Figure 1. Figure 2 shows the values of $\nu(s)$ and $\partial(s) = (\partial_1(s), \partial_2(s))$. These values are conveniently represented by a graph such as in Figure 3. The nodes represent different states of the circuit, with s and $\partial(s)$ shown for each of them. The edges of the graph lead from state s to states in $\nu(s)$. A possible behavior of the circuit must thus be a path in the graph.

To satisfy finite delay property, the circuit must not remain forever within a region of the graph where the state of some component is constant and has a constant tendency to change. For example, it cannot remain forever within the subset $\{00, 01\}$ of states: that would mean s_1 being forever 0 with a constant

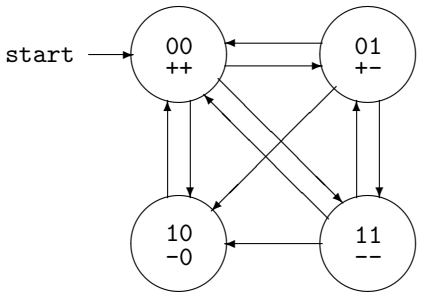


Figure 3: State graph for the circuit of Figure 1.

tendency to increase. Being in the region $\{00, 01\}$ is thus an unstable situation; the circuit must leave that region within a finite time. Other such "unstable regions" are:

$\{10, 11\}$ $\quad (s_1 = 1,\ \partial_1(s) = -)$;
$\{01, 11\}$ $\quad (s_2 = 1,\ \partial_2(s) = -)$.

The following sequences of states are examples of possible circuit behavior:

$00, 01, 11, 10, (00, 01, 11, 10)^\omega$;
$00, 11, (00, 11)^\omega$;
$00, 10, (00, 10)^\omega$.

Note that in the last example s_2 remains constant and equal to 0, although in the state 00 it has a tendency to change to 1. The reason for this being a possible behavior is that this tendency disappears in state 10. The following are examples of impossible behavior:

$11, 10, (11, 10)^\omega$;
$00, 01, (00, 01)^\omega$;
$00, 01, 11$.

(The first contains an impossible state succession, the remaining two violate finite delay property.)

3 Communicating sequential processes

We have a system of p sequential processes $P_1, P_2, \ldots, P_p$, communicating by means of v common variables $V_1, V_2, \ldots, V_v$. For $1 \leq k \leq p$, the process P_k can assume states from a finite set S_k. For $1 \leq k \leq v$, the variable V_k can assume states from a finite set S_{p+k}. The state of the system is described by an n-tuple $s = (s_1, s_2, \ldots, s_n)$ where $n = p + v$ and $s_k \in S_k$ is the state of P_k for $1 \leq k \leq p$, or the state of V_{k-p} for $p < k \leq n$. We denote the set of all these n-tuples by S.

Each process executes a sequence of *moves*. A move changes the state of the process executing it, and may also change the state of one or more common variables. The result depends on the state of the process and, possibly, the state of common variables. Several processes may move at the same time. If none of such simultaneous moves involves access to common variables, their results combine in the same way as for the circuits in Section 2. Otherwise, the result depends on the access mechanism. In some cases, this mechanism ensures that the moves are "atomic", that is, the result is always the same as if the moves were executed one after another, in any order. In other cases, it may inhibit simultaneous execution of certain moves. In general, the result of moves, individual or simultaneous, is described by a function $\nu : S \to 2^S$. We know that if a system state s changes, it can only change to a state from the set $\nu(s) \subseteq S$.

We do not know which process or processes are next to move in a given system state. We only know which of the following applies to each process and system state:

(1) The process operates with an unknown, but non-zero speed.
(2) The process operates with an unknown, possibly zero, speed.
(3) The process cannot move.

Case (1) is the normal situation. Case (2) occurs in problems where participating processes have an option to quit without affecting others. Case (3) means that the process reached its end state, or is waiting for some event. For $1 \leq k \leq p$, define:

$\partial_k(s) = +$ if P_k has a non-zero speed in system state s,
$\partial_k(s) = 0$ otherwise.

Define also $\partial_k(s) = 0$ for $p < k \leq n$ and $s \in S$. A precise formulation of (1)–(3) is: if $\partial_k(s) \neq 0$ for some s and k, either the value of ∂_k or the state of P_k must change within a finite time.

Example 2

The two processes in Figure 4, communicating by means of common variable V_1. This is one of incorrect solutions to the mutual exclusion problem presented in [2]. It appears as an example in [4]. We recall that a correct solution must guarantee three things, namely: "mutual exclusion" (the processes must never be in their critical sections at the same time), "non-blocking" (a process leaving its remainder of cycle must reach its critical section within a finite time), and "partial operability" (a process must be able to reach its critical section even if the other never leaves its remainder of cycle).

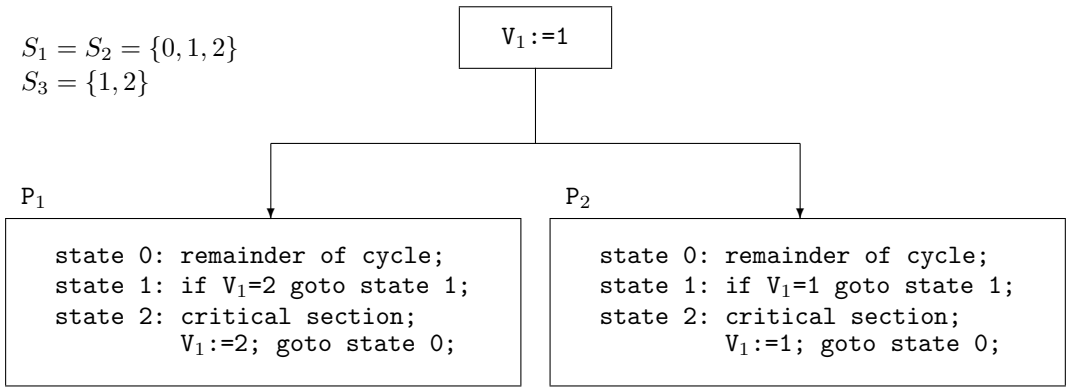


Figure 4: Two communicating processes.

s	$\nu(s)$	$\partial(s)$
001	{101,011,111}	000
002	{102,012,112}	000
011	{111}	000
012	{112,022,122}	0+0
021	{121,001,101}	0+0
022	{122,001,101}	0+0
101	{201,111,211}	+00
102	{112}	000
111	{211}	+00

s	$\nu(s)$	$\partial(s)$
112	{122}	0+0
121	{221,101,201}	++0
122	{101}	0+0
201	{002,211,012}	+00
202	{002,212,012}	+00
211	{012}	+00
212	{012,222,022}	++0
221	{201,022}	++0
222	{201,022}	++0

Figure 5: Values of ν and ∂ for the processes of Figure 4.

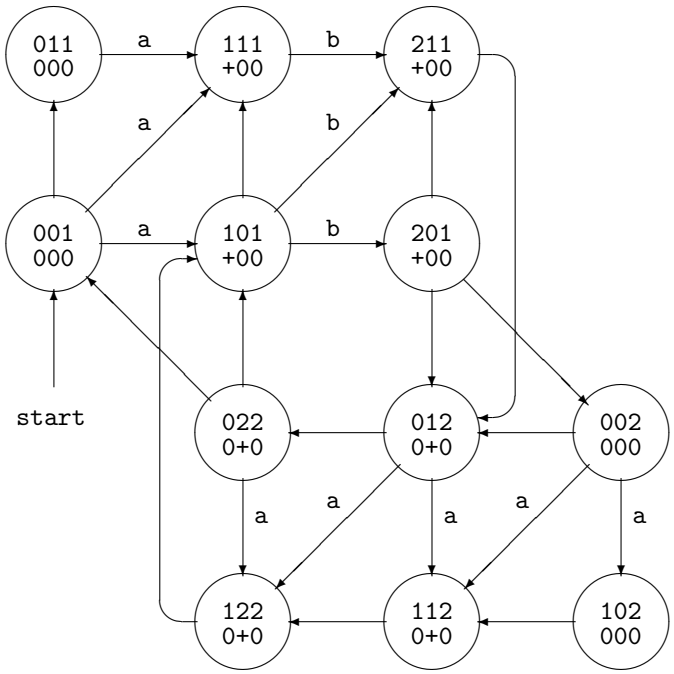


Figure 6: State graph for the processes of Figure 4.

Figure 5 shows the values of $\nu(s)$ and $\partial(s) = (\partial_1(s), \partial_2(s), \partial_3(s))$. The values of ν are as obtained in [4]. The short loop leading from state 1 back to 1 in system states $1x2$ and $x11$ is not considered a move because it does not change the process state. Simultaneous setting, or testing and setting, of V_1 by different processes is assumed impossible, so moves from system states 122, 211, and $22x$ are restricted to one process at a time.

The values of $\partial(s)$ represent assumptions about speeds of the processes. Thus, each process is assumed to have a non-zero speed while it is in state 2. This is implied by the whole problem: a process remaining forever within its critical section would block the entire system. The same applies to P_1 in system states $1x1$ and P_2 in system states $1x2$.

The requirement of partial operability means that a process is allowed to remain forever in state 0, so the processes are assumed to have an "unknown, possibly zero" speed in that state. As P_1 cannot move in system states $1x2$ and P_2 in system states $x11$, the tendency is 0 for these states.

The state graph of the system is shown in Figure 6 (its layout copied from [4]). It shows only the states accessible from $s = 001$. One can see that mutual exclusion is ensured: states of the form $22x$ are not accessible. As in Example 1, we can identify here the "unstable regions" – those with constant s_k and ∂_k:

$$\{101, 111\} \qquad (s_1 = 1,\ \partial_1(s) = +);$$
$$\{201, 211\} \qquad (s_1 = 2,\ \partial_1(s) = +);$$
$$\{012, 112\} \qquad (s_2 = 1,\ \partial_2(s) = +);$$
$$\{022, 122\} \qquad (s_2 = 2,\ \partial_2(s) = +).$$

The system must not remain forever in any of these regions. The following are examples of possible system behavior:

$$001;$$
$$(001, 111, 211, 012, 022)^{\omega};$$
$$001, 101, 201, 002, 102.$$

In the first, neither of the processes ever leaves its remainder of cycle. In the second, the processes repeatedly leave their remainder of cycle (at the same time), and are allowed to enter their critical sections

according to V_1. In the third, P_2 never leaves its remainder of cycle, and prevents P_1 from entering its critical section for the second time. The system remains forever in state 102; partial operability is not ensured.

Example 3

The two processes shown in Figure 7. This is another incorrect solution to the mutual exclusion problem from [2]. Figure 8 shows the values of $\nu(s)$ and $\partial(s)$.

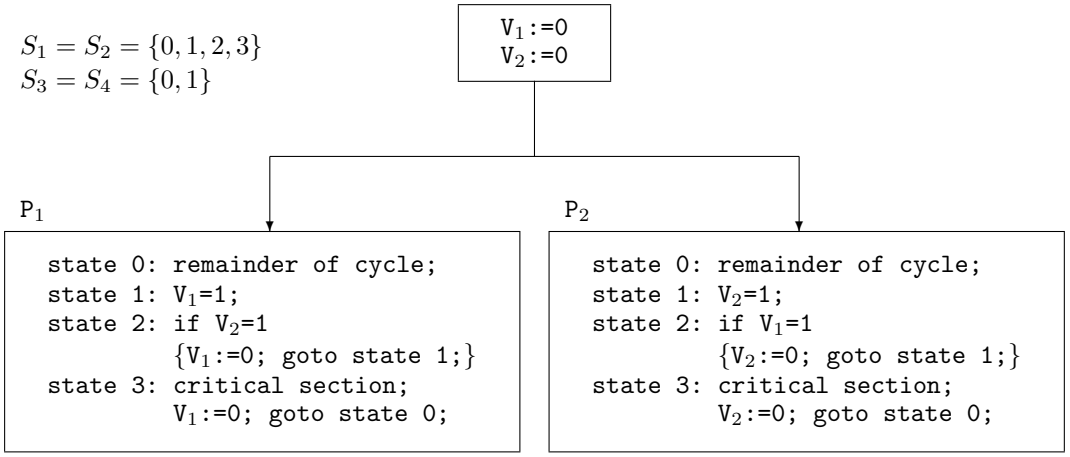


Figure 7: Two communicating processes.

s	$\nu(s)$	$\partial(s)$
0000	{1000,0100,1100}	0000
0100	{1100,0201,1201}	0+00
0201	{1201,0301,1301}	0+00
0301	{1301,0000,1000}	0+00
1000	{2010,1100,2110}	+000
1100	{2110,1201,2211}	++00
1201	{2211,1301,2311}	++00
1301	{2311,1000,2010}	++00

s	$\nu(s)$	$\partial(s)$
2010	{3010,2110,3110}	+000
2110	{3110,2211,3211}	++00
2211	{1201,2110,1100}	++00
2311	{1301,2010,1000}	++00
3010	{0000,3110,0100}	+000
3110	{0100,3211,0201}	++00
3211	{0201,3110,0100}	++00

Figure 8: Functions ν and ∂ for the processes of Figure 7.

The state graph of the system is shown in Figure 9. The vertical arcs are moves of P_1, the horizontal ones are moves of P_2. In order not to obscure the picture, simultaneous moves of P_1 and P_2 are not shown; they would all appear as "vector sums" of single-process moves. The Figure shows only the states accessible from $s = 0000$. One can see that the mutual exclusion is ensured: states $33xx$ are not accessible.

As in Example 2, we can identify here the "unstable regions":

$\{0100, 1100, 2110, 3110\}$ $\quad (s_1 = 1,\ \partial_1(s) = +);$
$\{0201, 1201, 2211, 3211\}$ $\quad (s_1 = 2,\ \partial_1(s) = +);$
$\{0301, 1301, 2311\}$ $\quad (s_1 = 3,\ \partial_1(s) = +);$
$\{1000, 1100, 1201, 1301\}$ $\quad (s_2 = 1,\ \partial_2(s) = +);$
$\{2010, 2110, 2211, 2311\}$ $\quad (s_2 = 2,\ \partial_2(s) = +);$
$\{3010, 3110, 3211\}$ $\quad (s_2 = 3,\ \partial_2(s) = +).$

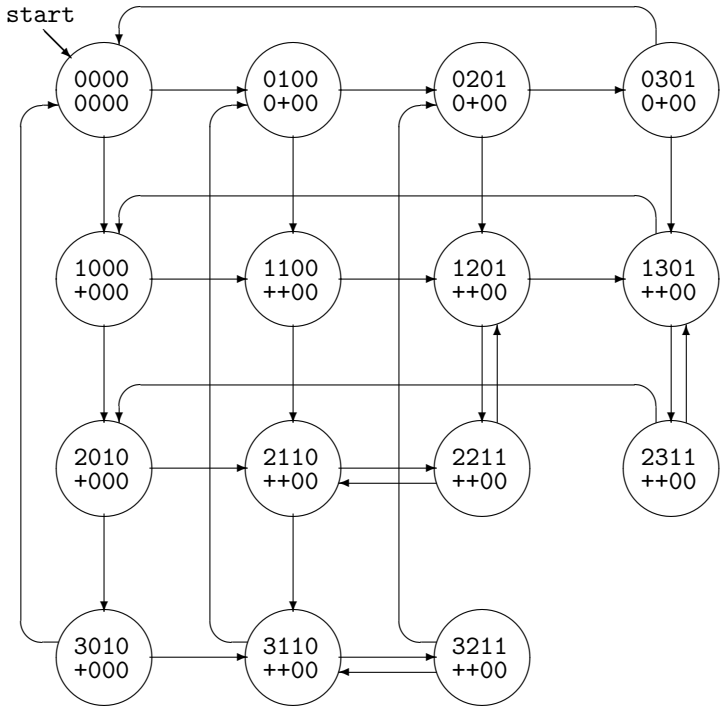


Figure 9: State graph for the processes of Figure 7. Simultaneous moves not shown.

The following are examples of possible behavior:

$(0000, 1000, 2010, 3010)^\omega$;

$0000, (0100, 1100, 1201, 1301, 1000, 1100, 2110, 3110)^\omega$;

$0000, 0100, 1100, (1201, 2211, 2110, 2211)^\omega$.

In the first, P_1 repeatedly vistis its critical section while P_2 does not move from remainder of cycle. In the second, both processes visit their critical sections. The third is the classical "after you – after you" situation. The processes repeat the same moves without ever reaching their critical sections; non-blocking is not ensured.

4 Common formulation

In each case above, we study a system of $n > 1$ components $C_1, C_2, \ldots, C_n$ (electronic devices, processes, commmon variables) whose states evolve in time in a mutually dependent manner, according to some laws. For $1 \leq k \leq n$, component C_k can assume states from a finite set S_k. The state of the system is described by an n-tuple $(s_1, s_2, \ldots, s_n) \in S = S_1 \times S_2 \times \cdots \times S_n$. At each moment, the state of C_k has a tendency to change that is described by an element of a finite set T_k. The set T_k may contain special element 0 meaning no tendency to change.

We have only a partial knowledge of laws governing the system's behavior. Our knowledge consists of n functions $\partial_k : S \to T$ and function $\nu : S \to 2^S$. We know that in system state s the tendency of C_k to change state is $\partial_k(s)$. We know that if this tendency is not 0, either the state of C_k, or its tendency to change, or both, will change within a finite time. We know further that if system state s state changes, it can only change to a state from the set $\nu(s)$.

The above description can be simplified if we consider the tendency to change as part of the component's state. For $1 \leq k \leq n$, let us redefine the *state of* C_k to be a pair (s_k, t_k) where $s_k \in S_k$ is the state of C_k in the old sense and $t_k \in T_k$ its tendency to change. Denote $S_k \times T_k$ by Q_k. Define $(s_k, t_k) \in Q_k$ to be *stable* if $t_k = 0$, and *unstable* otherwise.

Redefine *system state* to be an n-tuple $(q_1, q_2, \ldots, q_n)$ where $q_k \in Q_k$ for $1 \leq k \leq n$. For system state $q = ((s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n))$, let q_s and q_t denote, respectively, the n-tuples $(s_1, s_2, \ldots, s_n)$ and $(t_1, t_2, \ldots, t_n)$.

Define Q to be the set of system states q where $q_t = (\partial_1(q_s), \partial_2(q_s), \ldots, \partial_n(q_s))$. These are the system states that conform to functions ∂_k. For $q \in Q$, define $\mathrm{Next}(q)$ to be the set of all $q' \in Q$ such that $q'_s \in \nu(q_s)$. These are all the system states that can follow q.

We can now consider generally a system that assumes states from a given set $Q \subseteq Q_1 \times Q_2 \times \cdots \times Q_n$, where $Q_1, Q_2, \ldots, Q_n$ are finite sets. The elements of each Q_k for $1 \leq k \leq n$ are classified into stable and unstable. We know that if state q changes, it can only change to a state from the set $\mathrm{Next}(q)$, where $\mathrm{Next} : Q \to 2^Q$ is a given function. We do not know how this state is determined, but we know that any unstable component q_k of system state must change within a finite time.

The study of the system consists of a thought-experiment where we start the system in a given initial state, and then observe it for an infinitely long period of time. We note the initial state, and then the new state every time the state changes. The resulting sequence of states is called a *run* of the system. A run can be finite or not; a finite run represents the situation where, from some time on, the state does not change any more. A run r is thus a sequence (finite or not) of elements from Q. We denote its i-th element by $r(i)$. The k-th component of $r(i)$ is denoted by $r_k(i)$. The set of all indices in r is denoted by $\mathrm{Dom}(r)$.

Because of our partial knowledge, we cannot predict the exact run resulting from the experiment. We can only classify the runs into possible, that is, consistent with our knowledge, and impossible, that is, inconsistent with it. A sequence r of states represents a possible run if and only if it satisfies these two conditions:

(a) $r(i+1) \in \mathrm{Next}(r(i))$ whenever $i, i+1 \in \mathrm{Dom}(r)$;
(b) For $1 \leq k \leq n$ and $i \in \mathrm{Dom}(r)$, if $r_k(i)$ is unstable, there exists $j \in \mathrm{Dom}(r)$, $j > i$, such that $r_k(j) \neq r_k(i)$.

Condition (a) is equivalent to that of r being a path in an inputless, nondeterministic automaton with the set of states Q and transition function $\mathrm{Next} : Q \to 2^Q$. The state graph of this automaton for Examples 1, 2, and 3 is that shown, respectively, in Figures 3, 6 and 9.

Condition (b) classifies the paths into possible and impossible. Being expressed in terms of components of state, it is rather difficult to visuslize. Below, we express it more directly in terms of the state graph.

Define an *unstable region* to be any set of the form $U(k, u) = \{q \in Q \mid q_k = u\}$ where $1 \leq k \leq n$ and $u \in Q_k$ is unstable. Denote the family of all unstable regions by by $\mathbf{U}$. Define $\mathrm{In}(r)$ to be the set of all states that occur infinitely many times in a run r.

Proposition 1. *An infinite run r satisfies (b) if and only* $\mathrm{In}(r) \not\subseteq U$ *for each* $U \in \mathbf{U}$.

Proof. Let r be an infinite run. Assume that $\mathrm{In}(r) \not\subseteq U$ for each $U \in \mathbf{U}$. Consider any k and i such that $r_k(i)$ is unstable. By definition, $r_k(i) \in U(k, r_k(i))$. By assumption, some state not in $U(k, r_k(i))$ occurs infinitely many times in r. Thus, for some $j > i$ we have $r(j) \notin U(k, r_k(i))$, meaning $r_k(j) \neq r_k(i)$.

Assume now that r satisfies (b). Consider any $U = U(k, u) \in \mathbf{U}$. Suppose $r(i) \in U$ for some i. That means $r_k(i) = u$ is unstable. By (b), there exists $j > i$ such that $r_k(j) \neq r_k(i)$, meaning $r(j) \notin U(k, u)$. Any element of r that belongs to U is thus followed, sooner or later, by an element not in U. That means infinitely many elements of r are not in U. As Q is finite, at least one of them must repeat infinitely many times, thus belonging to $\mathrm{In}(r)$. □

Stated in another way, an infinite path in the state graph satisfies (b) if and only if it repeatedly visits the complement $Q - U$ of each $U \in \mathbf{U}$. One can think of the path as being attracted by each of the subsets $A = Q - U$ for $U \in \mathbf{U}$, so we can call these subsets *attractors*. Let $\mathbf{A}$ be the family of all such attractors. Proposition 1 can now be paraphrased as saying that r satisfies (b) if and only if it repeatedly visits each attractor. From this follows:

Corollary 1. *An infinite run satisfies (b) if and only if it belongs to the set*

$$(Q^* A_1 Q^* A_2 Q^* \ldots Q^* A_m)^\omega$$

where $A_1, A_2, \ldots, A_m$ are all the members of $\mathbf{A}$.

Let a subset of Q be called *terminal* if it has a nonempty intersection with each attractor in $\mathbf{A}$. Define $\mathbf{T}$ to be the family of all such terminal subsets. From the above discussion follows:

Corollary 2. *An infinite run r satisfies (b) if and only if* $\mathrm{In}(r) \in \mathbf{T}$.

Define state $q \in Q$ to be *final* if all its components q_k for $1 \leq k \leq n$ are stable. Denote the set of all final states by F. Notice that F is the intersection of all attractors, and that the singleton members of $\mathbf{T}$ are exactly those $\{q\}$ where $q \in F$.

Proposition 2. *A finite run satisfies (b) if and only if it ends with a final state.*

Proof. Consider any finite run $r = r(1), r(2), \ldots, r(p)$ and assume, in turn, that $r(p) \in F$ and $r(p) \notin F$. It is easy to see that (b) is satisfied in the first case and violated in the second. □

To summarize: the set of possible runs is identical to the set of paths in a nondeterministic automaton that satisfy certain condition. This condition can be alternatively expressed in terms of unstable regions, attractors, or terminal subsets.

The condition expressed in terms of terminal subsets has been introduced by Muller in [5], and an automaton with such condition has been since known as the "Muller automaton". The condition expressed in terms of attractors has been used in [8].

In most cases, we are not interested in all details of a run, but only in occurrences of few selected events (such as, for example, leaving the remainder of cycle and entering the critical section in Examples 2 and 3). We can suppress all the unwanted information by providing the automaton with output. We label the interesting us transitions with symbols from some finite alphabet $\mathcal{A}$, and all the remaining transitions with the empty word ε. In the usual way, we define the language generated by the automaton as the set of all words obtained by concatenating labels along the allowed paths starting at a given state. This language can always be expressed in a finite manner using the operations of union, product, star, and omega. It can be effectively computed using, for example, transition matrices and Corollary 1. The resulting expression contains the interesting us information in a condensed form.

Take, for instance, the system of Example 2. Suppose we are interested only in P_1 leaving its remainder of cycle and P_1 entering its critical section. We represent these events, respectively, by the outputs "a" and "b" as shown in Figure 6 (the ε-labels are omitted). The generated language is $(ab)^* \cup (ab)^\omega \cup (ab)(ab)^* a$. It contains words ending with "a", showing that P_1 may leave its remainder of cycle but never reach the critical section.

This approach makes it possible to verify the presence or absence of blocking – a version of finite delay on the system's scale – in a single and exhaustive argument. On may note that Gilbert and Chandler [4], who stopped short of introducing finite delay and arriving at Muller automaton, had to consider three different types of blocking with a separate argument for each. The problem with Muller automaton model is that it grows exponentially with the number of components and states. One can always reduce the model by considering only the states accessible form the initial state, and noticing that $\mathrm{In}(r)$ must be a strongly connected subset of the state graph. This, however, cannot compensate the exponential growth.

5 Final remarks

As we have seen, asynchronous circuits and communicating processes are similar to the extent that they can be described in the same way with the help of functions ν and ∂_k. This common description can be converted into more abstract one that uses next state function and unstable components of state. This, in turn, can be replaced by a nondeterministic automaton with behavior restricted by unstable regions, attractors, or terminal subsets.

Any differences between circuits and processes that may appear in these descriptions reflect the differences in which the way ν and ∂_k are defined. In the case of circuits, the result of multiple components changing state at the same time is always the union of idividual changes. In the case of processes, this is true only in the absence of communication via common variables. The effect of simultaneous moves that involve communication depends on the mechanism used for that communication and must be individually specified.

In the case of circuits, the tendency to change state and the possibility to do so are just two aspects of the same thing. This connection is weaker in the case of processes. In particular, a process may have no

tendency to move even if it has a possible move. (However, this assumption of "unknown, possibly zero speed" is often absent from models using finite delay property.)

Whenever these differences are not essential, it should be possible to exploit the similarity by extending results from one area of research to the other.

Acknowledgement

The author is indebted to Antoni Mazurkiewicz for a significant simplification to Section 4.

References

[1] Dijkstra, E. W. Cooperating sequential processes. Tech. Rep. EWD123, Technological University, Eindhoven, 1965.

[2] Dijkstra, E. W. Cooperating sequential processes. In *Programming Languages*, F. Genuys, Ed. Academic Press, New York, 1968, pp. 43–112.

[3] Farwer, B. ω-automata. In *Automata, Logics, and Infinite Games*, E. Grädel, W. Thomas, and T. Wilke, Eds., no. 2500 in Lecture Notes in Computer Science. Springer, 2002, pp. 3–21.

[4] Gilbert, P., and Chandler, W. J. Interference between communicating parallel processes. *Commun. ACM 15* (1972), 427–437.

[5] Muller, D. E. Infinite sequences and finite machines. In *Switching Circuit Theory and Logical Design: Proc. 4th Ann. Symp.* (1963), IEEE, New York, pp. 3–16.

[6] Muller, D. E., and Bartky, W. S. A theory of asynchronous circuits. In *Proc. Int. Symp. on the Theory of Switching, Part I* (1959), Harvard University Press, pp. 204–243.

[7] Perrin, D., and Pin, J.-E. *Infinite Words. Automata, Semigroups, Logic and Games*. No. 141 in Pure and Applied Mathematics. Elsevier, 2004.

[8] Redziejowski, R. R. The theory of general events and its application to parallel programming. Tech. Rep. TP 18.220, IBM Nordic Laboratory, Lidingö, 1972.

[9] Thomas, W. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol B: Formal Models and Semantics*, J. van Leeuwen, Ed. Elsevier Science Publishers, 1990, pp. 133–192.